Chaire en macroéconomie et prévisions

To Bag is to Prune

Philippe Goulet Coulombe[1]

Cahier de recherche
Working paper
2021-03

Mars / March 2021

ESG UQÀM
École des sciences de la gestion
Université du Québec à Montréal

[1] University of Pennsylvania

Chaire en macroéconomie et prévisions

La Chaire en macroéconomie et prévisions est fière de s'appuyer sur un partenariat avec les organisations suivantes:

Finances Québec

Hydro Québec

CDPQ

Les opinions et analyses contenues dans les cahiers de recherche de la Chaire ne peuvent en aucun cas être attribuées aux partenaires ni à la Chaire elle-même et elles n'engagent que leurs auteurs.

Opinions and analyses contained in the Chair's working papers cannot be attributed to the Chair or its partners and are the sole responsibility of the authors.

To Bag is to Prune

Philippe Goulet Coulombe*
Université du Québec à Montréal

First Draft: July 10, 2020
This Draft: June 8, 2021
Latest Draft Here

Abstract

It is notoriously difficult to build a bad Random Forest (RF). Concurrently, RF blatantly overfits in-sample without any apparent consequence out-of-sample. Standard arguments, like the classic bias-variance trade-off or double descent, cannot rationalize this paradox. I propose a new explanation: bootstrap aggregation and model perturbation as implemented by RF automatically *prune* a latent "true" tree. More generally, randomized ensembles of greedily optimized learners implicitly perform optimal early stopping *out-of-sample*. So there is no need to tune the stopping point. By construction, novel variants of Boosting and MARS are also eligible for automatic tuning. I empirically demonstrate the property, with simulated and real data, by reporting that these new completely overfitting ensembles perform similarly to their tuned counterparts — or better.

Keywords: Random Forest, Trees, Pruning, Greedy Algorithms, Double Descent, Deep Learning

*Département des Sciences Économiques, gouletc@sas.upenn.edu. For helpful discussions and/or comments on an earlier draft, I would like to thank Karun Adusumilli, Edvard Bakhitov, Frank Diebold, Maximilian Göbel, Samuel Gyetvay, Stephen Milborrow, Frank Schorfheide, Dalibor Stevanovic, David Wigglesworth and Boyuan Zhang. For excellent research assistance, I thank Hugo Couture, Akshay Malhotra and Tony Liu.

1 Introduction

Random Forest (RF) is a very stubborn benchmark in Machine Learning (ML) applications to tabular data, especially in economics. It can successfully predict asset prices (Gu et al., 2020), house prices (Mullainathan and Spiess, 2017), and macroeconomic aggregates (Medeiros et al., 2019; Chen et al., 2019; Goulet Coulombe et al., 2019). It can infer treatment effect heterogeneity (Athey et al., 2019), and estimate generalized time-varying parameters (Goulet Coulombe, 2020). The list goes on. But what makes it so infallible? To answer that question, and eventually understand the reasons behind RF's growing list of successful applications, it is better to start with an apparent paradox.

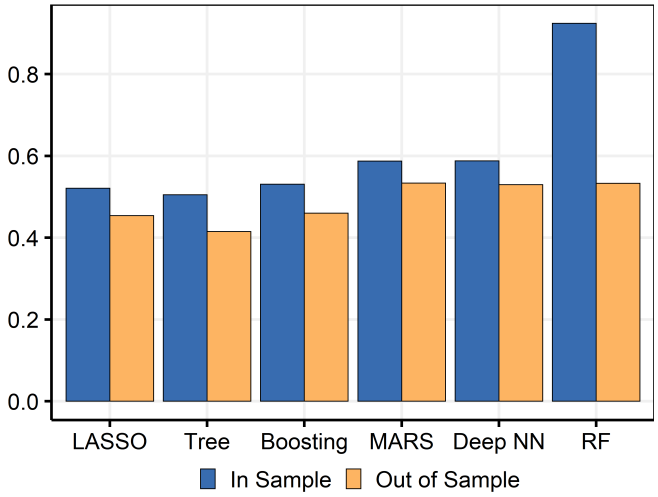


Figure 1: *Abalone* data set: comparing R^2_{train} and R^2_{test}. First four models hyperparameters are tuned by 5-fold CV. RF uses default parameters. NN details are in Appendix A.5.

Common statistical wisdom suggests that a non-overfitting supervised learning algorithm should have approximately the same mean squared error in the training sample as in the test sample. LASSO, Splines, Boosting, most Neural Networks (NN), and Multivariate Adaptive Regression Splines (MARS) abide by that principle. But not Random Forest. RF typically has an exceptionally high in-sample R^2 with a much lower, yet competitive, out-of-sample one. This means not only do the individual trees overfit the training set, but that the ensemble does, too. In contrast, the algorithms mentioned above usually perform poorly in such conditions. When optimally tuned, they are expected to deliver neighboring R^2_{test} and R^2_{train}. Figure 1 testifies to all those observations. This paper is about understanding why RF is excused from obeying the $R^2_{\text{test}} \approx R^2_{\text{train}}$ rule — and showing how to leverage this property for other algorithms.

Providing a theoretical reason to believe RF will not overfit, Breiman (2001) shows that the generalization error is bounded. That bound goes down as the individual learners' strength increases and goes up as correlation between them increases. Despite recent theoretical advances, like proving consistency (Scornet et al., 2015), it is still unclear why RF works so well on so many data

sets. It is acknowledged that much of that resilience is attributable to RF providing a flexible non-linear function approximator that does not overfit. Most importantly, unlike many models of the nonparametric family, the latter characteristic seems guaranteed even without resorting to careful hyperparameters tuning. Yet, it is still not clear what mechanism is behind this phenomenon.

If RF – made of fully grown completely overfitting trees – does not overfit out-of-sample, where does regularization come from? Clearly, increasing λ brings regularization in a ridge regression by shrinking coefficients toward zero, lowering the individual importance of each predictor. When it comes to RF, what contortions on the intrinsic model does its regularization entail? An appealing answer is that bagging smooths hard-thresholding rules (Bühlmann et al., 2002), like increasing the smoothness parameter of smoothing splines. If that were the whole story, RF, as does smoothing splines, would yield comparable R^2_{test} and R^2_{train}. Model averaging arguments would also have a similar implication.[1] As clearly displayed in Figure 1, it is not the case — so something else must be at work.[2] The newly proposed answer is: to bag (and perturb) is to prune.

More generally, I argue that randomized greedy optimization performs optimal early stopping. This is interesting since greedy optimization is often introduced in statistical learning books as an inevitable (but suboptimal) practical approach in the face of computational adversity (Friedman et al., 2001). It turns out the necessary evil has unsuspected benefits. A greedy algorithm treats what has already happened as given and what comes next as if it will never happen. While this depiction usually means "trouble", it is the key to this paper's argument. By recursively fitting a model and *not* re-evaluating what came before as the algorithm progresses, the work of early stages will be immune to subsequent overfitting steps, provided the latter averages out efficiently. Mechanically, when running CART, the structure at the top cannot be weakened by the bottom's doings – the bottom's existence is not even considered when estimating the top. Moreover, when faced with only noise left to fit in a terminal node, it is shown that a *Perfectly Random Forest*'s test set prediction is the sample mean, which is unbiased and – most importantly – has minimal variance. In short, it performs *pruning*.

Fortunately, not only trees are eligible for the enviable property, but also other greedily fitted additive models like Boosting and MARS. Based on this observation, I propose *Booging* and *MARSquake* which – like RF – are ensembles (of bagged and perturbed base learners) that completely overfit the training sample and yet perform nicely on the test set. Those are later shown to be promising alternatives to Boosting and MARS (both with a tuned stopping point) on real and simulated data sets. An `R` package implements both.

Finally, it is worth contrasting this paper's explanation with recent "interpolating regime" and

[1] This renders incomplete (at best) arguments linking RF regularization to that of penalized regression (originally discussed in Friedman et al. (2001), and more recently Mentch and Zhou (2019)) using results developed for *globally optimized* linear models (Elliott et al., 2013; LeJeune et al., 2020).

[2] Mullainathan and Spiess (2017)'s Table 1 – reporting results from off-the-shelf ML algorithms applied to house price prediction – is another convenient example where all aspects of the phenomenon are visible.

"double descent" ideas proposed to explain the success of deep learning (Belkin et al., 2019a,b; Hastie et al., 2019; Bartlett et al., 2020; Kobak et al., 2020). In a regression context, the interpolating regime is entered whenever one fits an algorithm of ever-increasing complexity past the complexity level delivering $R^2_{\text{train}} = 1$.[3] The double descent is the astonishing observation that for large-scale deep neural networks (DNN), the out-of-sample performance starts to increase past the point where $R^2_{\text{train}} = 1$. Preceded by the typical U-shaped empirical risk curve implied the classical bias–variance trade-off before $R^2_{\text{train}} = 1$, this makes it for a "double descent" — the first starting from $R^2_{\text{train}} = 0$ and the second from $R^2_{\text{train}} = 1$. Belkin et al. (2019a) evoke that the phenomenon is also present in RF. However, their construction mistakenly associates the number of trees to be increasing complexity (as in Boosting) whereas it is explicit increased averaging/regularization in RF. Thus, there is no double descent in RF, but rather a single *monotonic* descent. Section 2.3 elaborates on this matter. Wyner et al. (2017) also argue that interpolation may be the key for Boosted Trees and RF success because local fitting of dissident data points prevents harming the overall prediction function $\hat{f}$. But it is unclear as to why RF is so proficient at it, why "locality" emerges in the first place, and why estimation variance does not spread. The current paper makes exactly clear how the (greedy) construction of RF guarantees that overfitting washes away out-of-sample.

This paper is organized as follows. In section 2, I present the main insights and discuss their implications for RF and other greedy algorithms. In section 3, I demonstrate by means of simulations the implicit optimal early stopping property of RF, Booging and MARSquake. Section 4 applies the paper's main ideas to classic regression data sets. Section 5 concludes.

2 Randomized Greedy Optimization & Optimal Early Stopping

It is common to see that RF will have R^2_{train} magnitudes higher than R^2_{test}, a symptom which would suggest overfitting for many standard algorithms. That is, the traditionally defined in-sample fitted values $\hat{y}_i^{\text{RF}} = {}^1/_B \sum_{b=1}^{B} \hat{y}_{i,b}^{\text{tree}}$ where B is the total number of base learners, and corresponding residuals have nothing to do with what one gets when applying the estimated model to new data, unless the "true" R^2 is really high. While this R^2_{train} curiosity is usually of limited interest *per se*, it creates some intriguing headaches from a more traditional statistical perspective. For instance, any attempt to interpret the intrinsic RF *model* relies on measurements obtained on pseudo hold-out samples (called *out-of-bag*). In contrast, one would not refrain from exploring the structure of MARS' fitted values or that of a single tree. Indeed, most algorithms, when properly tuned, will produce comparable R^2_{test} and R^2_{train}. This implies that using the in-sample conditional mean $\hat{y}_i$ for any subsequent analysis is perfectly fine. In that way, they behave similarly to any classical nonparametric estimators where a bandwidth parameter must be chosen to balance estimation flexibility and the

[3] Interpolation means training data points are effectively interpolated by the fitted function $\hat{f}$ when $R^2_{\text{train}} = 1$.

threat of overfitting. Once it is chosen according to CV or some information criteria, in-sample values provide reliable estimates of the *true* conditional mean and error term.

I argue that RF's notably different behavior can be explained by the combination of two elements: greedy optimization and randomization of the recursive model fitting sequence. By construction, the instability of trees makes the latter an easy task: simply bootstrapping the original data can generate substantially different predictors (Breiman, 1996). The former, greedy optimization, is usually seen as the suboptimal yet inevitable approach when solving for a global solution is computationally unthinkable. In this section, I argue that greedy optimization, when combined with randomization of the model building pass, has an additional benefit. When combined in a properly randomized ensemble, no harm will come in letting each greedily optimized base learner completely overfit the training sample. In the case of RF, this translates to the heuristic recommendation of considering fully grown trees where each terminal node contains either a single observation or very few. Subsequently, those observations are leveraged to develop new algorithms inheriting RF's desirable properties.

2.1 What Happens in the Overfitting Zone Stays in the Overfitting Zone

In a global estimation procedure, overfitting will weaken the whole prediction function. More concretely, estimating many useless coefficients in a linear regression will inflate the generalization error by increasing the variance of *both* the few useful coefficients and the useless ones. Bagging such a model will still be largely suboptimal: the ensemble will still rely on an average of coefficients which are largely inferior to those that would be obtained from regression excluding the useless regressors. Hence, we are still in the standard case where $R^2_{\text{test}} < R^2_{\text{train}}$ reveals that the model's performance is inferior to that of an optimally pruned counterpart.

A greedily optimized model works differently. At each step of the forward pass, everything that came before is treated as given and what comes next as if it will never happen. That is, as the algorithm progresses past a certain step s, the function estimated before s is treated as given. And everything before s was estimated assuming anything past s to be non-existent. Eventually, the greedy algorithm will reach s^* where the only thing left to fit is the unshrinkable "true" error $\epsilon_i = \hat{\epsilon}_{i,s^*} = y_i - \hat{f}_{s^*}(x_i)$. The key is that entering deep in the overfitting zone will not alter $\hat{f}_{s-1}$ since it is not re-evaluated. As a result, early non-overfitting steps can be immune to the weakening effect of subsequent ones, as long as the latter efficiently averages out to 0 in the hold-out sample. An immediate implication of this separability property is that there is no need to stop the forward pass at the unknown s^* to obtain predictions immune to estimation variance inflation.

These abstract principles can be readily applied to think about fitting trees where a step s is splitting the subsample obtained from step $s-1$. A tree does not distinguish whether the current sample to split is the original data set of the result of an already busy sequence of splits. Moreover,

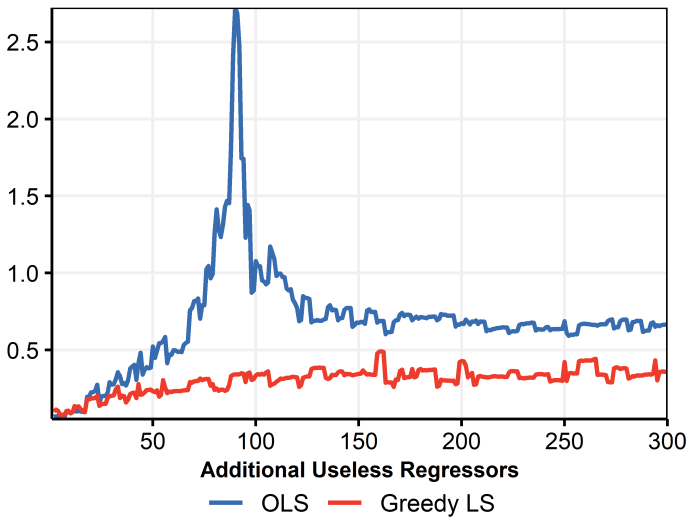


Figure 2: Model averaging/bagging different base learners with increasingly many useless features. Units are $\ln\left(\text{MSE}_{\text{model}}/\text{MSE}_{\text{Oracle}}\right)$. Oracle has 10 regressors, SNR=2, and $N = 100$. Details in Appendix A.2.

like any splits along the tree path, those optimized before venturing past s^* cannot be subsequently revoked. This implies that the predictive structure attached to them cannot be altered nor weaken by ulterior decisions the greedy algorithm makes.

Alternatively, we can think of fitting a linear regression with orthogonal features. A step s is adding a regressor by fitting it to the residual of the previous step. In this linear boosting case, we can hope that important predictors go in very soon in the process and are followed by many useless predictors until those are exhausted. Unlike the coefficients from the kitchen-sink OLS, the early fitted coefficients in the forward pass of the stagewise algorithm were estimated as part of a model that only included a handful of predictors. Those are precluded from the eventual weakening effect that comes with the inversion of a near-singular $X'X$.[4] Figure 2 supports those observations: for the same linear model, the effects of model averaging and bagging can differ substantially. Clearly, Greedy LS responds much better to ensembling than OLS in an environment incorporating noise and useless regressors. OLS's performance past the interpolation threshold ($R^2_{\text{train}} = 1$, which occurs at 90) is still order of magnitudes worse than Greedy LS (the graph is in log scale). There are limited benefits from tuning (moving along the x-axis) in Greedy LS, while those are huge for OLS.

2.2 Bagging and Perturbing as an Approximation to Population Sampling

At s^*, which corresponds to the *true* terminal node in the case of a tree, the DGP is simply

$$y_i = \mu + \epsilon_i. \tag{1}$$

[4]Adding a ridge penalty will alleviate the singularity problems, but will also (potentially heavily) shrink the real coefficients of interest, compromising their predictive power.

Clearly, the best possible prediction is the mean of all observations contained in the node. I argue that perfect randomization will also procure this optimal prediction out-of-sample, even if the ensemble itself is completely overfitting in-sample. This *Perfectly* Random Forest is, of course, merely a theoretical device and how close RF gets to this hypothetical version is an empirical question. Nevertheless, it is widely believed (and further confirmed in section 3) that bagging (`B`) and perturbing (`P`, draw `mtry` eligible features for a each split) trees can get very close to what would obtain from population sampling. By the latter, I mean that each tree is grown on non-overlapping samples from a population. Essentially, this is what bootstrapping any statistic is meant to approximate. Of course, the performance of any model will improve when averaging it over many close-to independent samples. The more subtle point being made here is that a good approximation to population sampling (via `B` & `P`) can generate a model whose structure will be close to the optimally pruned one, and that, without attempting any form of early stopping whatsoever. In other words, (1) more generally represents the truth from the hypothetical point s^* where a recursive fitting algorithm should optimally stop. Proper inner randomization assures that a prediction close to $\bar{y}$ is returned.

When faced with only noise left to fit, the prediction of a `B` & `P` ensemble of completely overfitting trees achieving perfect randomization is the (optimal) sample average. This result is obtained from taking the recursive view and assuming perfect randomization. First, the out-of-sample prediction of a RF for observation j is

$$\hat{\mu}_j^{\text{RF}} = {}^1/_B \sum_{b=1}^{B} \hat{\mu}_{j,b}$$

where $\hat{\mu}_{j,b}$ is the prediction of the b tree for observation j. Importantly, observation j is not included when fitting the trees, so we are looking at the prediction for a new data point using a function trained on observations $i \neq j$. For simplicity, assume fully grown trees, which means terminal nodes include a single observation.[5] The model is applied to the terminal node DGP in (1). Since the tree is fitting noise, perfect randomization implies that each out-of-sample tree prediction is a randomly chosen y_i for each b. The prediction is thus

$$\hat{\mu}_j^{\text{RF}} = \frac{1}{B} \sum_{b=1}^{B} y_{i(b)}.$$

Define $r = {}^B/_N$ where N is the number of training observations and r will eventually stand for "replicas". Since the $y_{i(b)}$'s amount to random draws of $y_{1:N}$, for a large enough B, we know with certainty that the vector to be averaged ($y_{i(1:B)}$) will contain r times the same observation y_i. Hence,

[5] This also directly implies that each base learners' R^2_{train} is one and that of the ensemble is bounded below by the subsampling rate, which will inevitably be much higher than $R^2_{\text{test}} = 0$.

the prediction equivalently is

$$\hat{\mu}_j^{\text{RF}} = \frac{1}{B}\sum_{i=1}^{N}\sum_{r'=1}^{r} y_{i,r'} = \frac{1}{B}\sum_{i=1}^{N}\sum_{r'=1}^{r} y_i = \frac{r}{B}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N} y_i$$

since $r = {}^{B}/_{N}$. In words, when a Perfectly Random Forest is starting to fit pure noise, its out-of-sample prediction averages out to the simple mean, which is *optimal* under (1) and a squared loss function. Intuitively, at s^*, the test set behavior of the prediction function (from fully grown trees) is identical to that of doing (random) subsampling with subsamples containing one observation. Averaging the results of the latter (over a large B) is just a complicated way to compute a mean. Hence, the out-of-sample prediction as provided by the perfectly random forest is one where implicit/automatic pruning was performed.[6] It is equivalent to that of an algorithm which knows the "true" s^*. A direct implication is that we need not to worry about finding s^* through cross-validation, since the optimally stopped prediction is what is being reported out-of-sample. Of course, this relies on a satisfying randomization level to be empirically attainable. Section 3 asks "How close to population sampling are we when fitting `B` & `P` trees?" and the answer is "surprisingly close".

The above also helps in understanding $R^2_{\text{test}} < R^2_{\text{train}}$ in RF. The gap's existence is a direct implication of implicit pruning via `B` & `P` being only active *out-of-sample*. Population sampling itself does not generate $R^2_{\text{test}} < R^2_{\text{train}}$[7], only its approximation by `B` & `P` does. A central role in this is that of `mtry` – the number of randomly selected features to be considered for a split. Overfitting situations can be thought of as an overabundance of parameters vs observations. The attached predictors are either directly available in the data or created via some form of basis expansions which trees is one (successful) possibility out of many. In such high-dimensional situations, it is clear that the model itself – the predictive structure – is barely identified: many different tree structure can rationalize a training sample with $R^2_{\text{train}} = 1$. Yet, these structures' predictions substantially differ when feeding in new data. This property of overfitting models (combined with the recursive fitting procedure) is the channel through which `mtry` strongly regularize the hold-out sample prediction. However, the resulting heterogeneity cannot deflate R^2_{train} since *different* overfitting base learners, when trained on the same data, provide the *same* fitted values (y_{train} itself).[8] Ergo, $R^2_{\text{test}} < R^2_{\text{train}}$.

[6] This provides a justification for Duroux and Scornet (2016)'s finding that pruning the base learners while shutting down `B` can deliver a performance similar to that of RF (provided a wise choice of tuning parameters).

[7] As the subsampling rate mechanically decrease – a luxury obtained from a growing sample size and fixed model complexity – the R^2_{train} itself will look much like a true R^2_{test} since the contribution of observation i to its out-of-bag prediction shrinks with subsampling rate.

[8] Thus, the training error remaining at 0 does not prevent *explicit* regularization from increasing, which is happening past $R^2_{\text{train}} = 1$ in Belkin et al. (2019a)'s RF example.

2.2.1 Not Your Average Model Averaging

At first sight, this may seem like nothing new: RF successfully controls overfitting by approximating more resampling by model averaging. The latter is known to provide a sort of regularization that can, in some special cases, be equivalent to more traditional shrinkage estimators (Elliott et al., 2013; LeJeune et al., 2020). What is new is that unlike averaging a kitchen-sink OLS regression (for instance), a greedy algorithm makes the structure estimated before s^* immune to what happens in the overfitting zone. In contrast, schemes like that of Eliasz et al. (2004) or those discussed in Rapach and Zhou (2013) imply directly or indirectly tuning the number of regressors in the base learner linear models. This means that including too many of them could damage the overall model's performance.[9] Of course, all of this is analogous to tuning λ in a ridge regression, and the resulting R^2_{test} is usually in the neighborhood of R^2_{train}.[10] Hence, `B` & `P` are not the source of the "paradox" *per se*: they must be paired with a greedy algorithm which can generate sufficient inner randomization.

Conversely, the ideas presented above could help at understanding why forecast combinations work so well, which unlike confirming their multiple successes, is still an ongoing venture (Timmermann, 2006). It is plausible that individual forecasters construct their predictive rule in an inductive recursive fashion. That is, human-based economic forecasting has likely more to do with a decision tree (based on looking at multiple time series plots) or a stepwise regression, than with the solution of a global problem (like OLS, LASSO and others). Indeed, it is arguably much easier to learn in a greedy fashion (both for a human and a computer) than to solve a complex multivariate problem directly for its global solution. Thus, assuming underlying forecasts are constructed as such, the average will behave in a very distinctive way if those are overfitting. As argued in Hellwig (2018) for the survey of IMF forecasters, the latter assertion is very likely true. As a result, the discussion above provides yet another explanation for the success of forecast combinations (especially the simple average scheme): significant inner randomization combined with recursively constructed overfitting forecasts provides implicit (and necessary) pruning. This is not a replacement but rather a complement to traditional explanations (usually for linear models) that link the effects of model averaging to traditional shrinkage estimators.[11]

[9] Figure 2 makes this distinction clear: OLS, while experiencing a reasonable performance renaissance thanks to double descent, could still strongly benefit from tuning the number of included regressors.

[10] This is why using a (global) linear model to think about `mtry`'s effect – while it may yield interesting insights (Mentch and Zhou, 2019) – provides an incomplete answer that fails to capture one of RF's most salient regularities: $R^2_{\text{train}} > R^2_{\text{test}}$.

[11] For instance, Friedman et al. (2001) discuss the link for RF itself, Rapach and Zhou (2013) discuss it for the case of forecasting stock returns with averages of linear models.

2.3 No Such Thing as Double Descent in Random Forest

In an influential paper, Belkin et al. (2019a) argue that Random Forest (RF), along with deep neural networks, exhibits a phenomenon now known as "double descent". Its occurrence for tree ensembles is an artifact of a strangely contstituted notion of function capacity. Belkin et al. (2019a) define it as increased tree depth until interpolation occurs, and augmenting the number of trees past that threshold. Unsurprisingly, the MSE starts to decrease sharply. The problem is, it does not do so because of increased functional complexity (like increasing the number of layers/neurons in a DNN, or regressors in a very large Ridge regression), but rather because *explicit* regularization (i.e., ensembling) has increased. More trees in RF is akin to simulating enough draws from a distribution of initial conditions so that the average model stabilizes, so it is not surprising to see that an ever-increasing number of trees brings down MSE until it reaches a plateau (Friedman et al., 2001) – it is the law of large numbers. RF may have more parameters than a single tree, but different trees are never used together to fit the same y_{train} (like in Boosting). As a result, averaging trees procuring $R^2_{\text{train}} < 1$ in isolation will *not* help the forest R^2_{train} climb to 1 — i.e., they do not increase function capacity. This is so because the additional parameters directly serve the purpose of model averaging (which can, in linear models, have similar effects to Ridge regularization, Elliott et al. 2013) rather than increased model complexity. Finally, Belkin et al. (2019a)'s U-shaped curve before $R^2_{\text{train}} = 1$ is expected because they use a single tree (similar curves are reported for plain CART in Figure 4), which is not a forest. Thus, with an appropriately defined notion of function capacity (underlying trees depth), there is no such thing as double descent of empirical risk for RF.[12] In fact, in line with this paper's ideas, we get something better: a single *monotonic* descent.

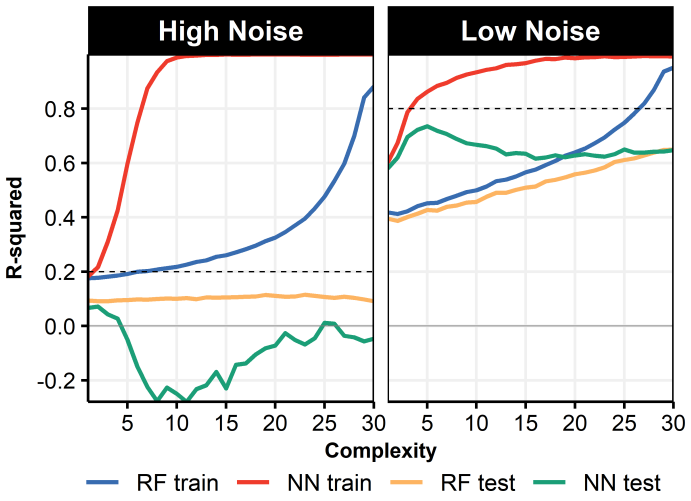


Figure 3: Dashed lines are true R^2. DGP is Friedman 1 (Friedman, 1991). The x-axis is an index of complexity/depth. For RF, it is a decreasing minimal size node from 200 to 1 in 30 steps, and for NN, an increasing number of layers from 1 to 30. The NN is 50 neurons wide and RF's `mtry` $= 1/3$.

[12]Moreover, constituent trees, the true modulator function capacity, cannot have more parameters than observations: once each observation is in its own cell, optimization is over.

Figure 3 compares NN and RF. In the high noise environment, even though NN out-of-sample performance can experience a revival in the overly parametrized regime, it remains far inferior to that of RF, which remains *constant* as its R^2_{train} goes way beyond the true R^2. Strikingly, while NN's depth needs to be tuned in each environment, the best tree depth is always the deepest one whether y is mostly noise or signal. Thus, when it comes to its crucial hyperparameter guiding function capacity (see blue line), there is no tuning problem in RF. Of course, this is an implication of the earlier discussion: explicit pruning or no pruning at all yield nearly identical performance.

It is not entirely surprising that RF behaves differently from NN, with their respective estimation being carried very differently. For NN (or Ridge) the number of parameters is fixed during the estimation/optimization whereas it is constantly evolving for a tree. That is, gradient descent methods typically estimate a model of fixed complexity globally. If that complexity level is too great, the model will fail to generalize. In contrast, for greedy methods, estimation/optimization and model building go hand in hand, and, importantly, different parameters are estimated in different stages within "transitory models" of varying complexity. Thus, it is no surprise that double descent does not transfer to RF, and that self-pruning is not expected from a NN.

2.4 Leveraging the Insight for Other Models

Not only trees require pruning. Many additive schemes must be optimally stopped at s^* to obtain the best test set performance. It has been discussed that mixing `B` & `P` with a greedy recursive algorithm can lead the algorithm to perform implicit optimal early stopping. It is natural to wonder if certain well-known greedy model building algorithms could also benefit for this property.

In sections 3 and 4, I consider `B` & `P` versions of MARS and Boosted Trees. Before jumping to do so, I discuss why they can plausibly benefit from it, but perhaps not as much as trees. The success of randomized greedy algorithms is bounded by the base learner's ability to generate a sufficiently diversified ensemble of predictors. If not, there will be benefits from stopping base learners earlier.

The reason why trees generate the most randomization among greedy methods is the *irreversibility* of the model building pass. Plain Boosting is used as the counterexample, but the principle clearly applies to MARS and similar greedily optimized additive models. Consider building a small symmetric tree of depth 2, the prediction function is

$$\hat{y}_i = I(x_i > 0)\left[\alpha_1 I(z_i > 0) + \alpha_2 I(z_i \leq 0)\right] + I(x_i \leq 0)\left[(\gamma_1 I(w_i > 0) + \gamma_2 I(w_i \leq 0)\right].$$

Finally, define $d^+_{x,i} = I(x_i > 0)$ as a regressor and the rest accordingly. We get

$$\hat{y}_i = \theta_1 d^+_{x,i} d^+_{z,i} + \theta_2 d^+_{x,i} d^-_{z,i} + \theta_3 d^-_{x,i} d^+_{w,i} + \theta_4 d^-_{x,i} d^-_{w,i}.$$

This representation shows trees are very singular "additive" models. It is clear that $d_{x,i}$ better be a good choice, because it is not going away: any term in the model building pass will be multiplied by it. By construction, no term added later in the expansion has the power to entirely undo the damage of a potentially harmful first split. In other words, splitting the sample is an *irreversible* action. This is what guarantees that steps occurring past s^* will not alter what was constructed before it.

Now, let us look at a toy boosting model where the base learners are single-split trees (stumps) and the learning rate is ν. An important difference with the above is that step s leading to

$$\hat{y}_i = \nu \left[\beta_1 d^+_{x,i} + \beta_2 d^-_{x,i}\right] + \cdots - \nu \left[\beta_1 d^+_{x,i} + \beta_2 d^-_{x,i}\right] \tag{2}$$

is absolutely possible. In words, by additivity, it is possible to correct any step that eventually turned out to be suboptimal in the search for a close-to global optima. With the randomization induced in Stochastic Gradient Boosting and other practical aspects, this is unlikely to happen exactly in those terms.[13] Nevertheless, a small ν and a large number of steps/trees in the additive model will mechanically increase the algorithm's potential for "reversibility". Indeed, Rosset et al. (2004) detail an equivalence between a procedure similar to the above and LASSO. If $\nu \to 0$, # of steps $\to \infty$ and regressors are uncorrelated, they obtain the LASSO solution – a *global* solution. Thus, there is an imminent tension between how close to a global optimization (2) can get and its capacity to generate inner randomization sufficiently to be dispensed from tuning the stopping point.

An interesting question is whether the properties detailed here apply to LASSO, which would free the world from ever tuning λ again. Indeed, when implemented via Least Angle Regression (Efron et al., 2004), the algorithm very much looks like a forward stagewise regression. In the spirit of the above, one would hope to let a randomized version of the regularization path roll until $\lambda = 0$, average those solutions and obtain the same R^2_{test} as if λ had been carefully tuned. Unfortunately, LASSO violates two of the requirements listed before. First, parameters are re-evaluated along the regularization path. For λ's that lay in the overfitting territory, the estimated coefficients will be weakened since they are re-estimated in an overcrowded model. Second, letting the model overfit (when $p < N$) implies setting $\lambda = 0$ which returns the OLS solution for any iteration, making the desired level of randomization likely unattainable.[14]

[13] Stochastic Gradient Boosting (Friedman, 2002) randomly selects a subset of observations at each step to train the weak learner.

[14] This last point could be alleviated, when in the $p > N$ case, the LASSO solution can include at most N predictors. In that scenario, the included set of variables would depend on the order within the regularization path (rather than its termination) which would increase randomization. Nevertheless, we cannot expect LASSO to benefit from automatic tuning because linear regression coefficients are re-evaluated along the estimation path.

2.5 Why RF is Not Equivalent to Pruning a Single Tree

Bagging and perturbing the model as implemented by RF leads to two enviable outcomes. The first is that the randomization procedure implicitly prunes an overfitting ensemble when applied to new data.[15] This was the subject of previous subsections. The second, more standard, is that as a result of randomization, RF performs orders of magnitude better than a single pruned true (Breiman, 1996). This is also observed in the simulations from section 3: `B` & `P` CART does much better than the ex-post optimally pruned base learner. In contrast, `B` & `P` MARS and Boosting will provide similar performance to that of their respective base learner stopped at s^*. Thus, RF must be pruning something else. I complete the argument of previous sections by arguing that its "pruning via inner randomization" is applied on the true *latent* tree $\mathcal{T}$ in

$$y_i = \mathcal{T}(X_i) + \epsilon_i \tag{3}$$

which itself can only be constructed from randomization. In short, it is the recursive fitting procedure itself that generates the need for Bagging.[16]

The inspiration for the following argument comes from forecasting with non-linear time series models, in particular with the so-called Self-Exciting Threshold Autoregression (SETAR). A simple illustrative SETAR DGP is

$$y_{t+1} = \eta_t \phi_1 y_t + (1 - \eta_t)\phi_2 y_t + \epsilon_t, \quad \eta_t = I(y_t > 0) \tag{4}$$

where ϵ_t is normally distributed. The forecasting problem consists in predicting y_{t+h} for $h = 1, ..., H$ given information at time t. As it is clear from (4), y_{t+1} is needed to obtain the *predictive function* for y_{t+2} which is either ϕ_1 or ϕ_2. Alas, only an estimate $\hat{y}_{t+1} = E(y_{t+1}|y_t)$ is available. By construction, $E(\hat{y}_{t+1}) = y_{t+1}$. However, by properties of expectations, $E(f(\hat{y}_{t+1})) \neq f(y_{t+1})$ if f is non-linear. Hence, proceeding to iterate forward using $\hat{y}_{t+h}$'s as substitutes for y_{t+h} at every step leads to a bias problem that only gets worse with the forecast horizon. If such an analogy were to be true for trees, this would mean that as the tree increase in depth, the more certain we can be that we are far from $\mathcal{T}(X_i)$, the optimal prediction function. I argue that it is the case.

Following the time series analogy, the prediction for a particular i can be obtained by a series of recursions. Define the cutting operator

$$\mathcal{C}(S; y, X, i) \equiv \mathcal{S}_i \left(\underset{k \in \mathcal{K}, c \in \mathbb{R}}{\arg\min} \left[\min_{\mu_1} \sum_{i \in \{S | X_k \leq c\}} (y_i - \mu_1)^2 + \min_{\mu_2} \sum_{i \in \{S | X_k > c\}} (y_i - \mu_2)^2 \right] \right)$$

[15] A simpler example is that of ridge regression: if there are more regressors than observations, then $\lambda \to 0$ leads to a multitude of solutions for β and the non-identification of the predictive structure.

[16] In Appendix A.6, I review a more standard case for Bagging based on presumed heteroscedasticity.

where $\mathcal{S}_i$ extract the subset that includes i out of the two produced by the splitting step. Inside the $\mathcal{S}_i$ operator is the traditional one-step tree problem. $\mathcal{K}$ is the set of potential features to operate the split at an optimized value c. $\boldsymbol{S}$ is the sample to split and is itself the result of previous cutting operations from steps $s-1$, $s-2$ and so on. To get the next finer subset that includes i, the operator is applied to the latest available subset: $\boldsymbol{S}' = \mathcal{C}(\boldsymbol{S}; y, X, i)$. The prediction for i can be obtained by using $\mathcal{C}$ recursively starting from $\boldsymbol{S}_0$ (the full data set) and taking the mean in the final $\boldsymbol{S}$ chosen by some stopping rule. In other words, the true tree prediction in (3) is $\mathcal{T}(X_i) = E\left(y_{i'} | i' \in \mathcal{C}^D(\boldsymbol{S}_0; y, X, i)\right)$ where D is the number of times the cutting operator must be applied to obtain the final subset in which i resides. To obtain the true tree prediction – the mean of observations in i's "true" terminal node – the sequence of $\mathcal{C}$'s must be perfect. Hence, consistency remains on safe ground: as the sample size grows large, estimation error vanishes and $\hat{\boldsymbol{S}} \to \boldsymbol{S}$ at each step. The finite sample story is, however, quite different.

Using $\hat{y}_{t+1}$ *in situ* of y_{t+1} in SETAR and $\hat{\boldsymbol{S}}$ *in situ* of $\boldsymbol{S}$ in a tree generate problems of the same nature. At each step, the expected composition of $\hat{\boldsymbol{S}}$ is indeed $\boldsymbol{S}$.[17] However, just like the recursive forecasting problem, the expected *terminal* subset is defined as an expectation over a recursion of nonlinear operators. Using $\hat{\boldsymbol{S}}$ rather than the unobserved $\boldsymbol{S}$ at each step does *not* deliver the desired expectation. Intuitively, getting the right k and c out of many possible combinations is unlikely. These small errors are reflected in $\hat{\boldsymbol{S}} \neq \boldsymbol{S}$ which is taken *as given* by the next step. Those errors eventually trickle down with absolutely no guarantee that they average out. In short, the direct CART procedure produces an unreliable estimate of a greedily constructed predictor $\mathcal{T}(X_i)$ because it takes as given at each step something that is not given, but estimated. Since $\mathcal{C}$ is a non-linear operator, this implies that the mean itself is not exempted from bias.

If the direct procedure cannot procure the right expected subset on which to take the average and predict, what will? The intuition for the answer, again, stems from the SETAR example. The proposed solution in the literature is – with a distinctively familiar sound – using bootstrap to simulate the intractable expectation (Clements and Smith (1997)).[18] $\hat{y}_{t+1}$ is augmented with a randomly drawn shock (from a parametric distribution or from those in the sample) and a forecast of y_{t+2} is computed conditional on it. Then, the procedure is repeated for B different shocks and the final forecast is the average of all predictions, which, by the non-linearity of $f()$, can make it a very different quantity from $f(\hat{y}_{t+1})$. A forecaster will naturally be interested in more than y_{t+2}. This procedure can be adapted by replacing the draw of a single shock by a series of them that will be used as the model is simulated forward. The prediction at step H is an average of forecasts at the end of each b randomly generated sequence.

Analogously, a natural approach is to simulate the distribution of $\boldsymbol{S}$ entering a next splitting step

[17] This notion can be formalized by defining the expectation in terms of indicator functions for each candidate observation. Each observation at each cutting step is expected to be classified in the right one of two groups.

[18] For a discussion of the SETAR case and other non-linear time series models, see section 2.7 in Khan (2015).

is to bootstrap the sample of the previous step, run $\mathcal{C}$ a total of B times, apply $\mathcal{C}$ in the next step and finally take the average of these B bootstrapped trees predictions. For a deeper tree, the growing process continues on the bootstrapped sample and the average is taken once the terminal condition is reached.

Coming to the original question: if RF is pruning something, what is it? I conjecture it is pruning $\mathcal{T}$ in (3). Unlike the implicit early-stopping property explained in section 2, this statement cannot be supported or refuted by the simulations presented in section 3. However, in Goulet Coulombe (2021), it is shown that under a "true tree" DGP, the performance of RF and a version of CART with a low learning rate coincides. The latter can be linked to fitting the true tree optimally via an (extremely) high-dimensional LASSO problem.

3 Simulations

Simulations are carried to display quantitatively the insights presented in the previous section. Namely, I want to display that **(i)** ideal population sampling of greedy algorithms performs pruning/early stopping, **(ii)** RF very closely approximates it for trees, and **(iii)** the property also extends to altered versions of Boosting and MARS.

3.1 Setup

I consider 3 versions of 3 algorithms on 5 DGPs. The 3 models are a single regression tree (CART), Stochastic Gradient Boosting (with tree base learners) and MARS. The five DGPs are a Tree, Friedman 1, 2 and 3 (Friedman (1991)) as well as a linear model (the sum of five mutually orthogonal and normally distributed regressors). The first two versions of each model are obvious. First, I include the plain model and second, a bootstrapped and perturbed ensemble of it, as described earlier. Additionally, `B` & `P` versions of MARS and Boosting have the so-called data augmentation (`DA`) option activated. It consists in enlarging the feature matrix to additionally incorporate $\tilde{X} = X + \mathcal{E}$ where $\mathcal{E}$ is a matrix of Gaussian noise. For categorical variables, $\tilde{X}$ is obtained by duplicating X and shuffling a fraction of its rows. Overall, `DA` can improve perturbation's potential when regressors are scarce. The Boosting and MARS `B` & `P` + `DA` versions will be referred to by the less gloomy-sounding sobriquets *Booging* and *MARSquake*. An `R` package implements both. Execution details are relegated to Appendix A.4.

The third version of each model, "Population Sampling" aims at displaying what results look like under the ideal case of perfect randomization. Subsampling is replaced by sampling B non-overlapping subsets of N observation from a population of $B \times N$ observations. This version help discern which algorithm generates enough inner randomization to get close to that desirable upper bound.

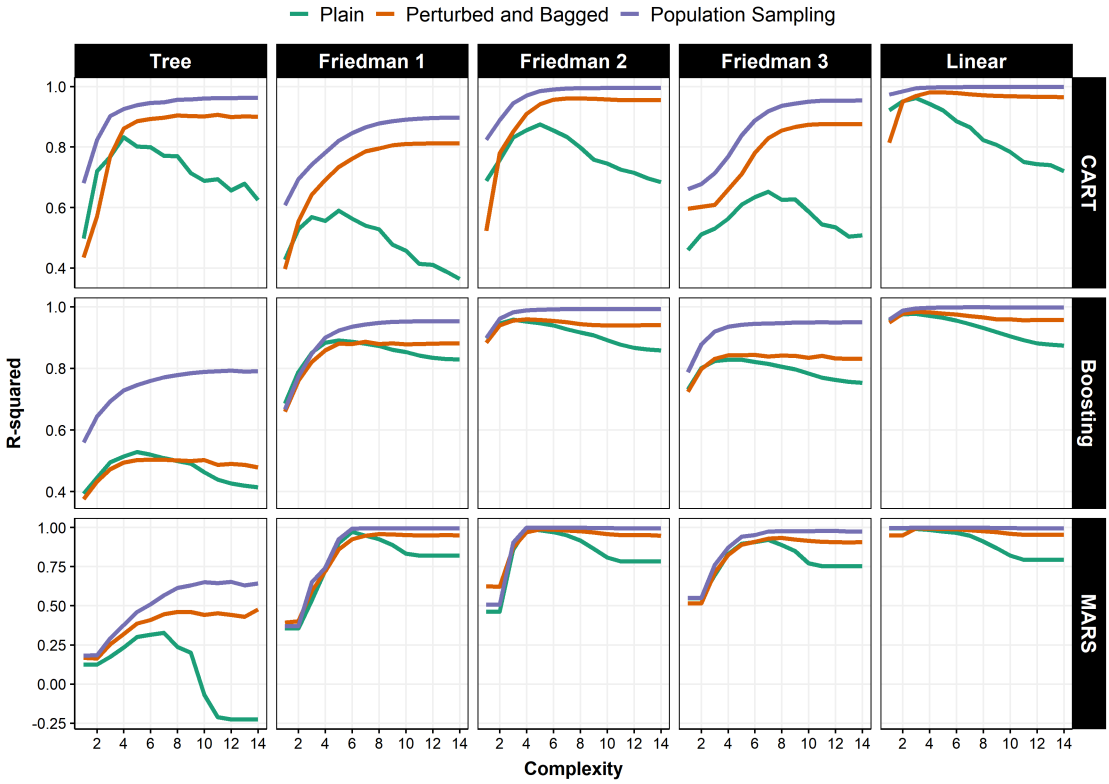


Figure 4: This plots the average hold-out sample R^2 between the prediction **and the true conditional mean** for 30 simulations. The level of noise is calibrated so the SNR is 4. Column facets are DGPs and row facets are base learners. The x-axis is an index of depth of the greedy model. For CART, it is a decreasing minimal size node $\in 1.4^{\{16,..,2\}}$, for Boosting, an increasing number of steps $\in 1.5^{\{4,..,18\}}$ and for MARS, it is an increasing number of included terms $\in 1.4^{\{2,..,16\}}$. Both training and test sets have $N = 400$.

3.2 Results

Figures 4 and 6 report the median R^2 between hold-out sample predictions and the true conditional mean for 30 simulations. Columns are DGPs and rows are models. The x-axis is an increasing index of complexity/depth for each greedy model. Overfitting should manifest itself by a decreasing R^2 past a certain depth. I consider two levels of noise, one that corresponds to a SNR ratio of 4 (Figure 4) and one of 1 (Figure 6 in Appendix). What does section 2 imply for the curves in those two Figures? First, the population sampling versions (purple line) should be weakly increasing since they perform implicit "perfect" early stopping. Second, the `B` & `P` versions (orange), should be parallel to those provided the underlying greedy model is generating enough inner randomization. Third, the value of the orange line at the point of maximal depth should be as high or higher than the maximal value of the green curve (i.e, the plain version's ex-post optimal stopping point).

When it comes to CART, those three properties are verified exactly. For any DGP, and both the population sampling and the `B` & `P` versions, increasing the complexity of the model by shrinking the minimal node size does *not* lead to a performance metric that eventually decrease. The striking parallelism of the purple and orange lines is due to trees generating enough inner randomization

with B & P so it performs self-pruning at a level comparable to that of the ideal experiment.[19] Depending on the DGP, the plain version follows a typical bias-variance trajectory: it generally follows the B & P one for some time before detaching itself from it past its ex-post optimal s^*. This early parallelism of the green and orange lines corroborate the idea that B & P CART (aka RF) performs implicit pruning.

Looking at Boosting and MARS, we again see that the population sampling line is weakly increasing in the respective depth of both models. If the B & P version fails to match this ideal shape, it is because the current specification cannot generate enough inner randomization. Figure 4 shows unequivocally encouraging results for both Boosting and MARS. For all DGPs, a clear pattern is observed: the B & P version's performance increases until it approximately reaches the optimal point (as can be ex-post determined by the hump in the green line) and then *remains at that level*, even if the base learners (one example being the 'Plain' version) are clearly suffering from overfitting. Under those conditions, it is fair to say that the enviable RF property is transferable to Boosting, and in a more pronounced fashion, MARS. When the noise level increase as depicted in Figure 6, we observed the same – albeit marginally less successful – phenomenon. Indeed, in those harder conditions, there is a small gap between ideal randomization and the one generated by Booging and MARSquake. However, the decrease in performance following the optimal depth is orders of magnitude smaller than what is observed for the plain version.

4 Empirics

This section has a more subtle aim than crowning the winner of a models' horse race. Rather than focusing on improving the tuned/pruned model which is already believed to be optimal, *Booging* and *MARSquake* bag and perturb completely overfitting based learners, which, as we will see, perform very poorly by themselves. Their performance will be compared to versions of Boosting and MARS where the optimal stopping point has been tuned by CV. The goal is to verify that in many instances, Booging and MARSquake provide similar predictive power to that of tuned models. Since CV's circumstantial imperfections are vastly documented (Krstajic et al., 2014; Bergmeir et al., 2018), it is not unrealistic to expect the B & P versions to sometimes outperform their tuned counterparts.

4.1 Setup

Most data sets are standard with a few additions which are thought to be of interest. For instance, many of the standard regression data sets have a limited number of features with respect to the number of observations. A less standard inclusion like *NBA Salary* has 483 observations and 26

[19] The purple line is mechanically expected to be at least above the orange one for a fixed depth: the former uses more data points which also helps at reducing estimation error.

features. *Crime Florida* pushes it much further with a total of 98 features and 90 observations. Those data sets are interesting because avoiding CV could generate larger payoffs in higher-dimensional setups. Still in the high-dimensional realm, but with the additional complication of non-*iid* data, are the 6 US macroeconomic data sets based on McCracken and Ng (2020).[20] Moreover, traditional CV can be overoptimistic in a time series context and avoiding it could help (Bergmeir et al., 2018). The 3 macroeconomic variables are quarterly GDP growth, unemployment change and inflation. I consider predicting those variables at an horizon of 1 quarter ($h = 1$) and 2 quarters ($h = 2$). Further information on data sets is gathered in Table 1. I do a 70-30 training-test split for all data sets.

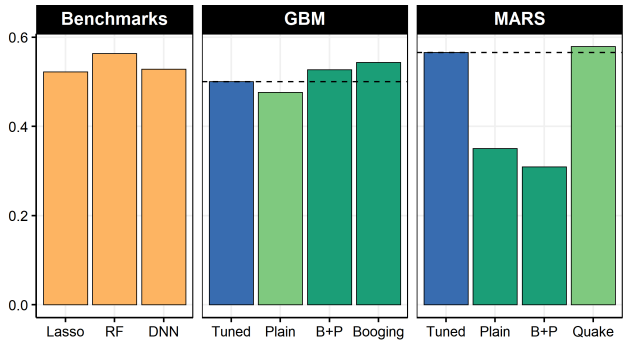
Beyond Boosting, MARS, and their novel variants, I include a few benchmark models. Those include LASSO, RF with default tuning parameters (`mtry`=1/3), a cost-complexity pruned regression tree, and two different neural networks (shallow and deep). Many additional NNs details are in Appendix A.5. For macro data sets, benchmarks additionally include an autoregressive model of order 2 (AR) and a factor-augmented regression with 2 lags (FA-AR) which are widely known to be hard to beat (Stock and Watson, 1999; Kotchoni et al., 2019; Goulet Coulombe et al., 2019). Remaining details needed to replicate results are in Appendix A.3.

4.2 Results

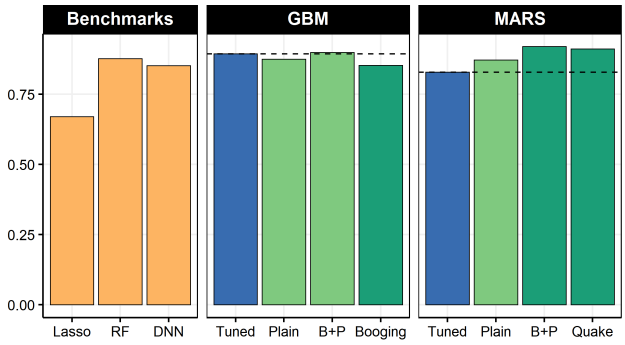
All prediction results are reported in Table 2 and an example is plotted in Figure 5. Moreover, to empirically document the R^2_{test} and R^2_{train} gap, Table 3 (Appendix) reports R^2_{train}'s. Overall, empirical results confirm the insights developed in section 2. In almost every instance, the overfitting ensembles do at least as well as the tuned version while completely overfitting the training sample, the same way RF would. Sometimes they do better. When they do not, going from `B` & `P` to MARSquake and Booging helps. This seldom occurring wedge suggests an important role for data augmentation when features are scarce.

For instance, on the *Abalone* data set, non-tuned MARS is overfitting, which leads to subpar performance. In line with simulation results, the newly proposed overfitting ensembles perform similarly well to using a single base learner and tuning it. Even better, Booging delivers statistically significant *gains* at the 1% confidence level. As RF, those two ensembles have a very high R^2_{train} (see Table 3 in the appendix) and yet, stellar performance is reported on the test set. For *Fish Toxicity*, *Red Wine*, *White Wine*, the plain overfitting versions are significantly worse than the tuned versions, and ensembling them delivers a performance (with respect to tuned counterparts) that is either significantly better or statistically indistinguishable. For *California Housing*, which has more than

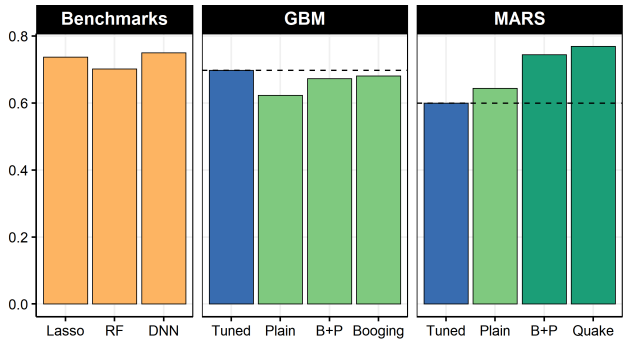
[20] Bagging has received attention of its own in the macroeconomic forecasting literature (Inoue and Kilian, 2008; Hillebrand and Medeiros, 2010; Hillebrand et al., 2020). However, nearly all studies consider the more common problem of variable selection via hard-thresholding rules – like t-tests (Lee et al., 2020). Those strategies are akin to what discussed in section 2.2.1, and cannot (and do not) strive for automatic pruning. Nevertheless, the motivation for using Bagging in their context is very close to what described for trees in section 2.5.



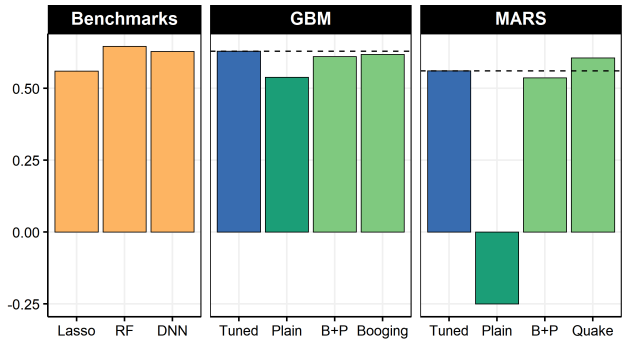
(a) *Abalone*



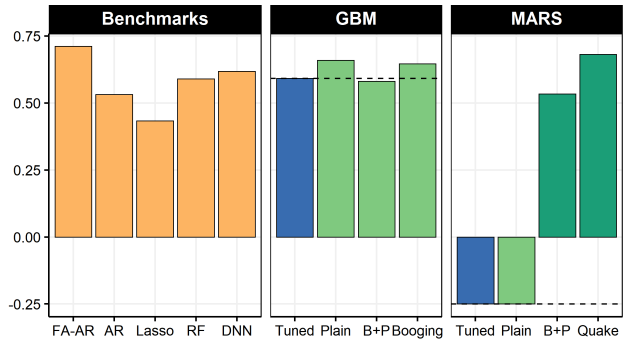
(b) *Boston Housing*



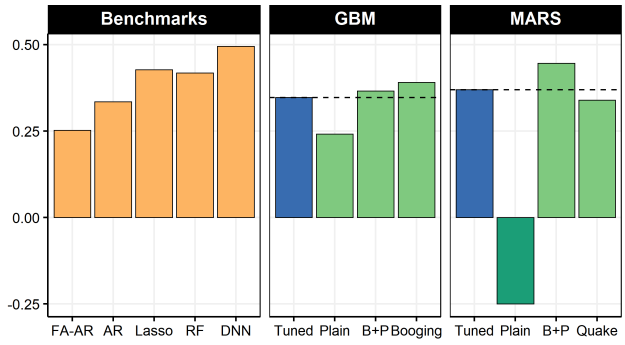
(c) *Crime Florida*



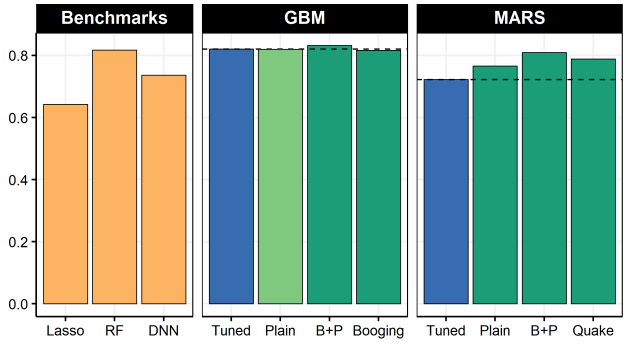
(d) *Fish Toxicity*



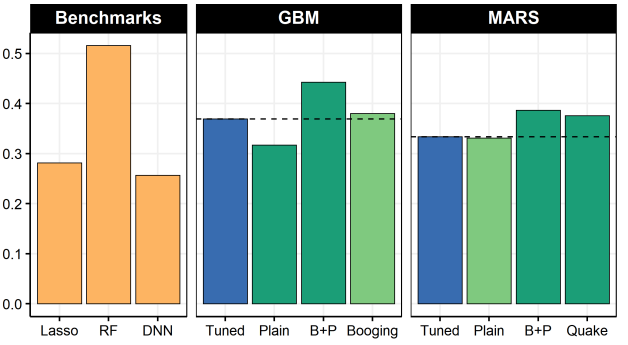
(e) *US Unemployment Rate* ($h = 1$)



(f) *US Inflation* ($h = 1$)



(g) *California Housing*



(h) *White Wine*

Figure 5: A Subset of Empirical Prediction Results. Performance metric R^2_{test}. Darker green bars means the performance differential between the tuned version and the three others is statistically significant at the 5% level using t-tests (and Diebold and Mariano (2002) tests for time series data). Light green means the difference is not significant at the prescribed level. To enhance visibility in certain cases, R^2_{test}'s below -0.25 are constrained to 0.25.

20 000 observations, all ensembles significantly improve over the tuned version for MARS.

For *Crime Florida* – the very high-dimensional case which is not time series – the two ensembles of completely overfitting MARS (their R^2_{train} are respectively 0.97 and 0.98) are doing much better than the tuned version. They both deliver a R^2_{test} of almost 0.8 in the case of MARSquake. The latter is also the overall second best model (being 1% less than NN) for this data set. Meanwhile, `B` & `P` Boosting and Booging are doing marginally better than the tuned version.

A now-familiar pattern is also visible for both unemployment and (to some extent) GDP at $h = 1$. Booging does as well as the tuned Boosting. Moreover, the former provides the best outcome among all models, with a 11% R^2_{test} increase with respect to both economic forecasting workhorses (AR, FA-AR). When it comes to plain and tuned MARS, all models are somewhat worse than the benchmarks with the tuned model itself delivering a terrible R^2_{test}. MARSquake is partially exempted from this failure for GDP, and completely is for unemployment. In the latter case, MARSquake is as good as FA-AR which incredible resilience is vastly documented (Stock and Watson, 2002; Goulet Coulombe et al., 2019). For inflation ($h = 1$), the best models are clearly `B` & `P` MARS and DNN. Finally, it is noteworthy that Booging dominates its tuned counterpart for all economic data sets. Thus, overfitting ensembles work well for economic forecasting where CV can be hazardous.

Lastly, on NN and Deep NN performances. DNN is mostly dominated by RF and other ensembles, with the exception of inflation where it narrowly beats `B` & `P` MARS. Giving the ongoing discussion on the properties of DNN's, it is interesting to check if DNN behaves similarly to RF. The short answer is "no". RF's R^2_{train} is *almost always* above 0.9, whereas that of DNN fluctuates highly depending on the target. Also, DNN does not display RF's emblematic resilience across data sets.

5 Conclusion

A fundamental problem is to detect at which point a learner stops learning and starts imitating. In ML, the common tool to prevent an algorithm from damaging its hold-out sample performance by overfitting is cross-validation. It is widespread knowledge that performing CV on Random Forests rarely yields dramatic improvements. Concurrently, it is often observed that $R^2_{\text{test}} < R^2_{\text{train}}$ without R^2_{test} being any less competitive. I argued that proper inner randomization as generated by bagging and perturbing the model, when combined with a greedy fitting procedure, will implicitly prune the learner once it starts fitting noise. By the virtues of recursive model building, the earlier fitting steps are immune to the instability brought upon by ulterior (and potentially harmful) steps. Once upon a time, the author heard a very senior data scientist and researcher say in a seminar, 'If you put a gun to my head and say "predict", I use Random Forest.' This paper rationalizes this feeling of security by noting that unlike other learners, RF performs its own pruning without the perils of cross-validation. Thus, it seems that, mixed with a proper amount of randomization, *greed is good*.

References

Athey, S., Tibshirani, J., Wager, S., et al. (2019). Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178.

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. (2020). Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019a). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

Belkin, M., Rakhlin, A., and Tsybakov, A. B. (2019b). Does data interpolation contradict statistical optimality? In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1611–1619. PMLR.

Bergmeir, C., Hyndman, R. J., and Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Bühlmann, P., Yu, B., et al. (2002). Analyzing bagging. *The Annals of Statistics*, 30(4):927–961.

Chen, J. C., Dunn, A., Hood, K. K., Driessen, A., and Batch, A. (2019). Off to the races: A comparison of machine learning and alternative data for predicting economic indicators. In *Big Data for 21st Century Economic Statistics*. University of Chicago Press.

Clements, M. P. and Smith, J. (1997). The performance of alternative forecasting methods for setar models. *International Journal of Forecasting*, 13(4):463–475.

d'Ascoli, S., Refinetti, M., Biroli, G., and Krzakala, F. (2020). Double trouble in double descent: Bias and variance (s) in the lazy regime. In *International Conference on Machine Learning*, pages 2280–2290. PMLR.

Diebold, F. X. and Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, 20(1):134–144.

Duroux, R. and Scornet, E. (2016). Impact of subsampling and pruning on random forests. *arXiv preprint arXiv:1603.04261*.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499.

Eliasz, P., Stock, J. H., and Watson, M. W. (2004). Optimal tests for reduced rank time variation in regression coefficients and for level variation in the multivariate local level model. *manuscript, Harvard University*.

Elliott, G., Gargano, A., and Timmermann, A. (2013). Complete subset regressions. *Journal of Econometrics*, 177(2):357–373.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378.

Geiger, M., Jacot, A., Spigler, S., Gabriel, F., Sagun, L., d'Ascoli, S., Biroli, G., Hongler, C., and Wyart, M. (2020). Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401.

Goulet Coulombe, P. (2020). The macroeconomy as a random forest. *arXiv preprint arXiv:2006.12724*.

Goulet Coulombe, P. (2021). Slow-growing trees. Technical report.

Goulet Coulombe, P., Leroux, M., Stevanovic, D., Surprenant, S., et al. (2019). How is machine learning useful for macroeconomic forecasting? Technical report, CIRANO.

Goulet Coulombe, P., Leroux, M., Stevanovic, D., Surprenant, S., et al. (2020). Prévision de l'activité économique au québec et au canada à l'aide des méthodes "machine learning". Technical report, Technical report, CIRANO.

Goulet Coulombe, P., Marcellino, M., and Stevanovic, D. (2021). Can machine learning catch the covid-19 recession? *CEPR Discussion Paper No. DP15867*.

Grandvalet, Y. (2004). Bagging equalizes influence. *Machine Learning*, 55(3):251–270.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. (2019). Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*.

Hellwig, K.-P. (2018). *Overfitting in Judgment-based Economic Forecasts: The Case of IMF Growth Projections*. International Monetary Fund.

Hillebrand, E., Lukas, M., Wei, W., et al. (2020). Bagging weak predictors. Technical report, Monash University, Department of Econometrics and Business Statistics.

Hillebrand, E. and Medeiros, M. C. (2010). The benefits of bagging for forecast models of realized volatility. *Econometric Reviews*, 29(5-6):571–593.

Inoue, A. and Kilian, L. (2008). How useful is bagging in forecasting economic time series? a case study of us consumer price inflation. *Journal of the American Statistical Association*, 103(482):511–522.

Khan, M. Y. (2015). *Advances in applied nonlinear time series modeling*. PhD thesis, lmu.

Kobak, D., Lomond, J., and Sanchez, B. (2020). The optimal ridge penalty for real-world high-dimensional data can be zero or negative due to the implicit ridge regularization. *Journal of Machine Learning Research*, 21(169):1–16.

Kotchoni, R., Leroux, M., and Stevanovic, D. (2019). Macroeconomic forecast accuracy in a data-rich environment. *Journal of Applied Econometrics*, 34(7):1050–1072.

Krstajic, D., Buturovic, L. J., Leahy, D. E., and Thomas, S. (2014). Cross-validation pitfalls when selecting and assessing regression and classification models. *Journal of cheminformatics*, 6(1):1–15.

Lee, T.-H., Ullah, A., and Wang, R. (2020). Bootstrap aggregating and random forest. In *Macroeconomic Forecasting in the Era of Big Data*, pages 389–429. Springer.

LeJeune, D., Javadi, H., and Baraniuk, R. (2020). The implicit regularization of ordinary least squares ensembles. In *International Conference on Artificial Intelligence and Statistics*, pages 3525–3535.

McCracken, M. and Ng, S. (2020). Fred-qd: A quarterly database for macroeconomic research. Technical report, National Bureau of Economic Research.

Medeiros, M. C., Vasconcelos, G. F., Veiga, Á., and Zilberman, E. (2019). Forecasting inflation in a data-rich environment: the benefits of machine learning methods. *Journal of Business & Economic Statistics*, (just-accepted):1–45.

Mentch, L. and Zhou, S. (2019). Randomization as regularization: A degrees of freedom explanation for random forest success. *arXiv preprint arXiv:1911.00190*.

Milborrow, S. (2018). *earth: Multivariate Adaptive Regression Splines*. R package.

Mullainathan, S. and Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106.

Olson, M. A. and Wyner, A. J. (2018). Making sense of random forest probabilities: a kernel perspective. *arXiv preprint arXiv:1812.05792*.

Rapach, D. and Zhou, G. (2013). Forecasting stock returns. In *Handbook of economic forecasting*, volume 2, pages 328–383. Elsevier.

Rasmussen, C. E. (1997). *Evaluation of Gaussian processes and other methods for non-linear regression*. PhD thesis, University of Toronto Toronto, Canada.

Rosset, S., Zhu, J., and Hastie, T. (2004). Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5(Aug):941–973.

Scornet, E., Biau, G., Vert, J.-P., et al. (2015). Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741.

Stock, J. H. and Watson, M. W. (1999). Forecasting inflation. *Journal of Monetary Economics*, 44(2):293–335.

Stock, J. H. and Watson, M. W. (2002). Macroeconomic forecasting using diffusion indexes. *Journal of Business & Economic Statistics*, 20(2):147–162.

Timmermann, A. (2006). Forecast combinations. *Handbook of economic forecasting*, 1:135–196.

Wyner, A. J., Olson, M., Bleich, J., and Mease, D. (2017). Explaining the success of adaboost and random forests as interpolating classifiers. *The Journal of Machine Learning Research*, 18(1):1558–1590.

A Appendix

A.1 Additional Graphs and Tables

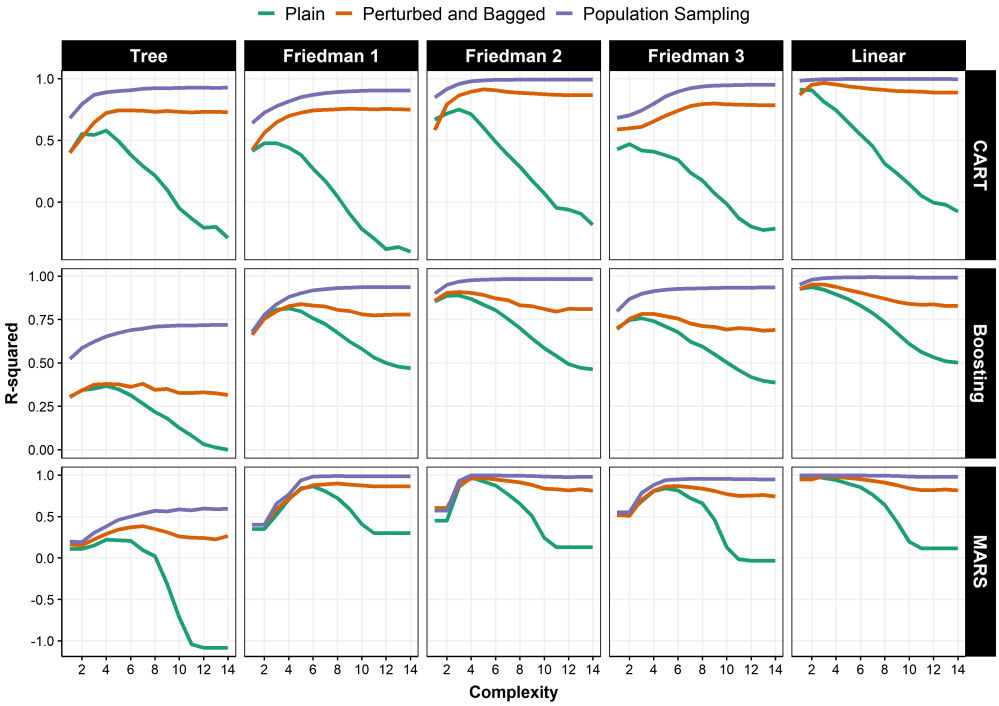


Figure 6: This plots the hold-out sample R^2 between the prediction and the true conditional mean. The level of noise is calibrated so the signal-to-noise ratio is 1. Column facets are DGPs and row facets are base learners. The x-axis is an index of depth of the greedy model. For CART, it is a decreasing minimal size node $\in 1.4^{\{16,..,2\}}$, for Boosting, an increasing number of steps $\in 1.5^{\{4,..,18\}}$ and for MARS, it is an increasing number of included terms $\in 1.4^{\{2,..,16\}}$.

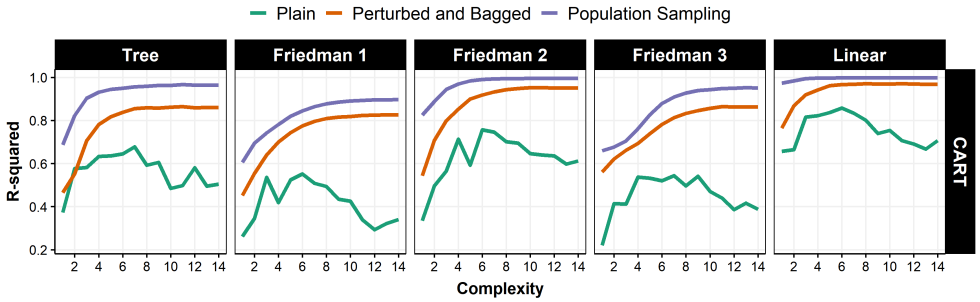


Figure 7: This is Figure 4's first row with `mtry`= 0.5.

Table 1: 20 Data Sets

Abbreviation	**Observations**	**Features**	**Data Source**
Abalone	4,177	7	archive.ics.uci.edu
Boston Housing	506	13	lib.stat.cmu.edu
Auto	392	7	archive.ics.uci.edu
Bike Sharing	17,379	13	archive.ics.uci.edu
White Wine	4,898	10	archive.ics.uci.edu
Red Wine	1,599	10	archive.ics.uci.edu
Concrete	1,030	8	archive.ics.uci.edu
Fish Toxicity	908	6	archive.ics.uci.edu
Forest Fire	517	12	archive.ics.uci.edu
NBA Salary	483	25	kaggle.com
CA Housing	20,428	9	kaggle.com
Crime Florida	90	97	census.gov
Friedman 1 R^2=.7	1,000	10	cran.r-project.org
Friedman 1 R^2=.4	1,000	10	cran.r-project.org
GDP h=1	212	599	Google Drive
GDP h=2	212	563	Google Drive
UNRATE h=1	212	619	Google Drive
UNRATE h=2	212	627	Google Drive
INF h=1	212	619	Google Drive
INF h=2	212	611	Google Drive

Notes: The number of features includes categorical variables expanded as multiple dummies and will thus be sometimes higher than what reported at data source website. Data source URLs are visibly abbreviated but lead directly to the exact data set or package being used. The number of features varies for each macro data set because a mild screening rule was implemented ex-ante, the latter helping to decrease computing time.

Table 2: R^2_{test} for all data sets and models

	Benchmarks							GBM				MARS			
	FA-AR	AR	LASSO	RF	Tree	NN	DNN	Tuned	Plain	B & P	Booging	Tuned	Plain	B & P	Quake
Abalone			0.52	0.56	0.45	0.54	0.53	0.50	0.48	0.53*	0.54**	0.57	0.35*	0.31*	**0.58*** **
Boston Housing			0.67	0.88	0.79	0.86	0.85	0.89	0.88	0.90	0.85*	0.83	0.87	**0.92**	0.91
Auto			0.66	**0.71**	0.61	0.13	0.64	0.64	0.59**	0.65	0.64*	0.71	−0.54 *	0.53	0.63
Bike Sharing			0.38	0.91	0.73	0.88	0.94	**0.95**	0.93***	0.91***	0.91***	0.71	0.89***	0.87***	0.90***
White Wine			0.28	**0.52**	0.28	0.37	0.26	0.37	0.32*	0.44***	0.38	0.33	0.33***	0.39**	0.38***
Red Wine			0.34	**0.47**	0.35	0.33	0.37	0.37	0.23**	0.37	0.38	0.38	0.29*	0.33	0.35
Concrete			0.59	0.90	0.71	0.89	0.88	**0.92**	0.92	0.90*	0.90***	0.83	0.87	0.30***	0.89
Fish Toxicity			0.56	**0.65**	0.57	0.60	0.63	0.63	0.54***	0.61	0.62	0.56	−0.25 ***	0.54*	0.61
Forest Fire			0.00	−0.11	0.00	−0.02	0.01	−0.03	−0.68 ***	−0.32 ***	−0.08	**0.01**	−1.55 *	−0.68	−0.36
NBA Salary			0.52	**0.60**	0.34	0.22	0.21	0.50	0.29***	0.49	0.50	0.36	0.11*	0.59*	0.53
CA Housing			0.64	0.82	0.59	0.75	0.74	0.82	0.82	**0.83*** **	0.82**	0.72	0.77***	0.81***	0.79***
Crime Florida			0.66	0.79	0.60	**0.82**	0.75	0.75	0.77	0.81*	0.79	0.70	0.44*	**0.81**	0.80
F1 $R^2 = 0.7$			0.53	0.62	0.50	0.43	0.51	0.65	0.54***	0.60***	0.67**	0.68	0.55	0.62	**0.69*** **
F1 $R^2 = 0.4$			0.32	0.40	0.36	0.19	0.28	0.40	0.16***	0.34*	0.41	**0.41**	0.14*	0.35	0.40*
GDP h=1	0.27	0.27	0.24	0.35	0.18	0.06	0.26	0.36	0.17	0.37	**0.38**	0.00	−9.08 ***	−0.45 **	−0.12 **
GDP h=2	−0.03	0.17	−0.01	0.16	0.00	−0.06	−0.52	0.15	−0.56 **	**0.20**	0.18	−0.40	−4.37 **	−0.41 *	−0.37 ***
UNRATE h=1	**0.71**	0.53	0.43	0.59	0.22	−0.69	0.62	0.59	0.66	0.58	0.65	−0.65	−0.72 ***	0.53	0.68
UNRATE h=2	**0.52**	0.29	0.26	0.37	0.16	0.14	0.41	0.43	0.35	0.42	0.48	0.16	−0.80 **	−0.28	0.26
INF h=1	0.25	0.33	0.43	0.42	0.25	0.41	**0.49**	0.35	0.24	0.37	0.39	0.37	−0.57 **	0.45	0.34
INF h=2	0.05	0.22	0.09	0.28	0.45	0.19	**0.51**	0.15	−0.26 ***	0.16	0.27*	0.39	−2.50 **	0.24	0.42

Notes: This table reports R^2_{test} for 20 data sets and different models, either standard or introduced in the text. For macroeconomic targets (the last 6 data sets), the set of benchmark models additionally includes an autoregressive model of order 2 (AR) and a factor-augmented regression with 2 lags (FA-AR). Numbers in bold identify the best predictive performance of the row. For GBM and MARS, t-test (and Diebold and Mariano (2002) tests for time series data) are performed to evaluate whether the difference in predictive performance between the tuned version and the remaining three models of each block is statistically significant. '*', '**' and '***' respectively refer to p-values below 5%, 1% and 0.1%. F1 means "Friedman 1" DGP of Friedman (1991).

Table 3: R^2_{train} for all data sets and models

	Benchmarks							GBM				MARS			
	FA-AR	AR	LASSO	RF	Tree	NN	DNN	Tuned	Plain	B & P	Booging	Tuned	Plain	B & P	Quake
Abalone			0.50	0.92	0.50	0.60	0.59	0.53	0.85	0.86	0.91	0.57	0.65	0.78	0.61
Boston Housing			0.72	0.98	0.87	0.90	0.89	1.00	1.00	0.99	0.99	0.90	0.97	0.97	0.98
Auto			0.68	0.96	0.77	0.13	0.81	0.86	1.00	0.98	0.98	0.77	0.98	0.93	0.96
Bike Sharing			0.38	0.98	0.89	0.95	0.96	0.95	0.94	0.95	0.71	0.89	0.88	0.90	
White Wine			0.26	0.92	0.27	0.47	0.75	0.44	0.82	0.85	0.88	0.37	0.46	0.52	0.51
Red Wine			0.29	0.91	0.41	0.40	0.42	0.41	0.96	0.94	0.95	0.44	0.56	0.69	0.67
Concrete			0.61	0.98	0.75	0.91	0.93	0.98	0.99	0.98	0.99	0.88	0.98	0.74	0.95
Fish Toxicity			0.54	0.93	0.60	0.64	0.61	0.92	0.97	0.95	0.97	0.63	0.96	0.82	0.88
Forest Fire			0.00	0.81	0.00	0.00	0.07	0.40	0.97	0.88	0.91	0.04	0.62	0.73	0.76
NBA Salary			0.47	0.93	0.72	0.65	0.71	0.99	1.00	0.97	0.97	0.64	0.92	0.84	0.93
CA Housing			0.63	0.97	0.61	0.78	0.85	0.86	0.89	0.91	0.90	0.72	0.80	0.83	0.81
Crime Florida			0.65	0.96	0.84	0.88	0.94	1.00	1.00	0.98	0.98	0.75	1.00	0.97	0.98
F1 $R^2 = 0.7$			0.45	0.93	0.45	0.62	0.71	0.95	1.00	0.97	0.97	0.65	0.81	0.84	0.86
F1 $R^2 = 0.4$			0.23	0.89	0.30	0.34	0.35	0.48	1.00	0.94	0.94	0.38	0.64	0.75	0.76
GDP h=1	0.41	0.11	0.23	0.91	0.51	0.26	0.44	0.81	1.00	0.96	0.96	0.47	1.00	0.94	0.94
GDP h=2	0.26	0.06	0.07	0.89	0.00	0.26	0.55	0.76	1.00	0.95	0.95	0.29	1.00	0.94	0.95
UNRATE h=1	0.57	0.40	0.48	0.93	0.81	-0.07	0.82	0.83	1.00	0.97	0.97	0.76	0.99	0.97	0.96
UNRATE h=2	0.41	0.13	0.35	0.92	0.38	0.42	0.25	0.99	1.00	0.96	0.96	0.75	1.00	0.96	0.96
INF h=1	0.76	0.73	0.90	0.97	0.81	0.64	0.94	1.00	1.00	0.99	0.99	0.73	1.00	0.99	0.99
INF h=2	0.69	0.63	0.72	0.96	0.72	0.67	0.92	1.00	1.00	0.99	0.98	0.81	1.00	0.99	0.98

Notes: This table reports R^2_{train} for 20 data sets and different models, either standard or introduced in the text. For macroeconomic targets (the last 6 data sets), the set of benchmark models additionally includes an autoregressive model of order 2 (AR) and a factor-augmented regression with 2 lags (FA-AR). F1 means "Friedman 1" DGP of Friedman (1991).

A.2 Simulations Details

For Figure 2, the true linear model has 10 mutually uncorrelated regressors, SNR=2, and $N = 100$. OLS has a slight built-in advantage because it always include the relevant regressors plus useless ones whereas Greedy LS has to select them itself. It is understood that OLS's performance would deteriorate even further without this crutch. Each prediction is the average over 50 models and 20 bagging replicas. For OLS, bagging is bypassed since it provides the same expectation as using the full sample once. The 50 models are constructed as follows: for each model, I generate x new useless regressors and add them to the relevant ones, then run estimation. The Greedy OLS is operationalized with the function `glmboost` in `R`, setting the learning rate at 1.

Figures 4 and 6. In terms of standard hyperparameters, Boosting has the shrinkage parameter $\nu = 0.1$, the fraction of randomly selected observations to build trees at each step is 0.5, and the interaction depth of those trees is 3. Of course, while those are fixed for all simulations, we will want to tune them once we get to real data. However, here, the point is rather to study the hold-out sample performance of each model as its depth increase, and compare that across the 3 versions. MARS has the polynomial degree set to 3.[21] RF is used with a rather high `mtry` of $9/10$ so to be better visually in sync with plain CART at a given depth.[22] The subsampling rate is $2/3$ for all bagged models.

The true tree DGP is generated using a CART algorithm's prediction function as a "new" conditional mean function from which to simulate. The "true" minimal node size being used is 40 (10% of the training set). Friedman DGP are obtained from the package `tgp` and typically generate a data set with 5 useful regressors (used in the true DGP) and 5 useless ones.

A.3 Empirics Details

For all data sets, I keep 70% of observations for training (and optimizing hyperparameters if needed) and the remaining 30% to evaluate performance. For cross-sectional data sets, those observations are chosen randomly. For time series applications, I keep the observations that consist of the first 70% in the sample as the training set. The test set starts before the 2001 recession and ends in 2014, which conveniently includes two recessions. Lastly, a seldomly binding outlier filter is implemented. Every prediction that is larger than twice the maximal absolute difference (in the training sample) with respect to the mean is replaced by the RF prediction (which is immune to outliers since it cannot extrapolate). This last addition is particularly helpful to prevent wildly negative R^2_{test} for non-tuned plain MARS and (less frequently) Boosting.

[21] For those unfamiliar with this machinery, see Friedman (2002) for Boosting and Friedman (1991) or Milborrow (2018) for MARS.

[22] For completeness, results when using `mtry`$= 1/2$ for both plain CART and RF are reported in Figure 7. It is clear that in the high signal-to-noise ratio environment, the milder perturbation of `mtry`$=9/10$ is preferable.

The X matrix for the macroeconomic data sets is based on Goulet Coulombe (2020)'s recommendations for ML algorithms when applied to macro data, which is itself a twist (for statistical efficiency and lessen computational demand) on well-accepted time series transformations (to achieve stationarity) as detailed in McCracken and Ng (2020).[23] Each data set has 212 observations and around 600 predictors. The number of features varies across macro data sets because a mild screening rule was implemented ex-ante, the latter helping to decrease computing time.

A.4 Implementation Details for *Booging* and *MARSquake*

Booging and MARSquake are the `B` & `P` +`DA` versions of Boosted Trees and MARS, respectively. The data-augmentation option will likely be redundant in high-dimensional situations where the available regressors already have a factor structure (like macroeconomic data).

ABOUT B. For both algorithms, `B` is made operational by subsampling. As usual, reasonable candidates for the sampling rate are $2/3$ and $3/4$. All ensembles use $B = 100$ subsamples.

ABOUT P. The primary source of perturbation in Booging is straightforward. Using subsamples to construct trees at each step is already integrated within Stochastic Gradient Boosting. By construction, it perturbs the Boosting fitting path and achieve a similar goal as that of the original `mtry` in RF. Note that, for fairness, this standard feature is also activated for any reported results on "plain" Boosting.

The implementation of `P` in MARSquake is more akin to that of RF. At each step of the forward pass, MASS evaluate all variables as potential candidates to enter a hinge function, and select the one which (greedily) maximize fit at this step. In the spirit of RF's `mtry`, `P` is applied by stochastically restricting the set of available features at each step. I set the fraction of randomly considered X's to $1/2$.

To further enhance perturbation in both algorithms, we can randomly drop a fraction of features from base learners' respective information sets. Since `DA` creates replicas of the data and keep some of its correlation structure, features are unlikely to be entirely dropped from a boosting run, provided the dropping rate is not too high. I suggest 20%. This can is analogous to `mtry`-like randomly select features, but for a whole tree (in RF) rather than at each split.

ABOUT DA. Perturbation work better if there is a lot to perturb. In many data sets, X is rich in observations but contains few regressors. To assure `P` meets its full randomization potential, a cheap data augmentation procedure can be carried. `DA` is simply adding fake regressors that are correlated with the original X and maintain in part their cross-correlation structure. Say X contains K regressors. I take the $N \times K$ matrix X and create two duplicates $\tilde{X} = X + \mathcal{E}$ where $\mathcal{E}$ is a matrix

[23] Goulet Coulombe et al. (2019) further study optimal data transformations for machine macroeconomic forecasting for many series and algorithms.

of Gaussian noise. SD is set to $1/3$ that of the variable. For X_k's that are either categorical or ordinal, I create the corresponding $\tilde{X}_k$ by taking X_k and shuffling 20% of its observations.

LAST WORD ON MARS. It is known that standard MARS has a forward and a backward pass. The latter's role is to prevent overfitting by (traditional) pruning. Obviously, there is no backward pass in MARSquake. Certain implementations of MARS (like *earth*, Milborrow (2018)) may contain foolproof features rendering the forward pass recalcitrant to blatantly overfit in certain situations (usually when regressor are not numerous). To partially circumvent this rare occurrence, one can run MARS again on residuals obtained from a first MARS run which failed to attain a high enough R^2_{train}.

A.5 Additional NN details

The first NNs is shallow (2 layers of 32 and 16 neurons) and is inspired from Gu et al. (2020). Such an architecture has provided reasonable performance on Canadian (Goulet Coulombe et al., 2020) and UK macroeconomic data Goulet Coulombe et al. (2021). The second is a deep NN (DNN, with 10 layers of 100 neurons) following the recommendations of Olson and Wyner (2018) for small data sets.

For both neural networks, the batch size is 32 and the optimizer is Adam (with Keras default values). Continuous X's are normalized so that all values are within the 0-1 range.

More precisely, **NN** in Table 2 is a standard feed-forward fully-connected network with an architecture in the vein of Gu et al. (2020). There are two hidden layers, the first with 32 neurons and the second with 16 neurons. The number of epochs is fixed at 100. The activation function is *ReLu* and that of the output layer is linear. The learning rate $\in \{0.001, 0.01\}$ and the LASSO λ parameter $\in \{0.001, 0.0001\}$ are chosen by 5-fold cross-validation. A batch normalization layer follows each *ReLu* layers. Early stopping is applied by stopping training whenever 20 epochs pass without any improvement of the cross-validation MSE.

More precisely, **DNN** in Table 2 is a standard feed-forward fully-connected network with an architecture closely following that of Olson and Wyner (2018) for small data sets. There are 10 hidden layers, each featuring 100 neurons. The number of epochs is fixed at 200. The activation function is *eLu* and that of the output layer is linear. The learning rate $\in \{0.001, 0.01, 0.1\}$ and the LASSO λ parameter $\in \{0.001, 0.00001\}$ are chosen by 5-fold cross-validation. No early stopping is applied.

A.6 Bagging and Heteroscedasticity

Grandvalet (2004) expands on Breiman (1996) and discuss in greater detail why bagging can boost trees' performance but not so much for OLS or splines. His argument basically boils down that trees

are non-linear functionals of the data while splines or OLS are just linear combinations of the data. In the case of OLS, perturbing the data weights B times gives a similar $\hat{\beta}$ as computing OLS with all weights being equal to 1. However, $\hat{E}^{\text{tree}}_{\boldsymbol{\omega}}(y|X)$ can be very far from just computing the same expectation at the mean $\omega_i = 1 \ \ \forall i$. Hence, if ω_i in

$$y_i = \mathcal{T}(X_i) + \omega_i \epsilon_i, \qquad \epsilon_i \sim N(0,1) \tag{A.1}$$

follows a certain non-degenerate distribution, it is argued that bagging will yield significant improvements. Of course, under these conditions (and a linear DGP), OLS would still be consistent, so that as the sample gets large, heteroscedascity does not compromise prediction.[24] That is, $\hat{E}^{\text{OLS}}(y|X;\boldsymbol{\omega}) \to \hat{E}^{\text{OLS}}(y|X;\hat{\boldsymbol{\omega}} = \mathbf{1})$ as the sample size grows. No such guarantees are available for complicated non-linear recursive estimators, such as trees.

Such reasoning can be extended to finite samples and in a straightforward application of a basic property of expectations: $E(f(\omega)) \neq f(E(\omega))$ unless f is linear in ω. If f is only mildly non-linear – like for the OLS or ridge functional, the shortcut $f(E(\omega))$ will be a reliable approximation to the real expectation of interest (Breiman (1996) refers to those as "stable" predictors). If $f = \mathcal{T}$, the shortcut likely provides an abysmal approximation. An alternative is to resort to "pairs" bootstrap (or subsampling) to implicitly simulate from a plausible distribution of ω_i and then use the mean over many bootstrapped trees to obtain $\hat{E}^{\text{tree}}_{\boldsymbol{\omega}}(y|X)$. Coming back to the main point of this paper, it is clear that pruning CART is an imperfect enterprise because the model it is pruning will not coincide to the true conditional expectation *if* ω_i's are heterogeneous.

Nevertheless, relying on presumed "badness" in the data to justify RF's usual supremacy over a single tree seems thin. There are many examples where heteroscedascity is visibly absent from the test set errors and yet, RF will do much better than (pruned) CART.

[24] A different story occurs in small samples where down-weighting noisy observations can provide substantial improvements. One example out of many is the use of stochastic volatility to improve (even) point forecasts in a macroeconomic context.